Puzzle

rs@toprllc.com July 13, 2020

Introduction

In this projectlet, we study the classic **Game of Fifteen** (`https://en.wikipedia.org/wiki/15_puzzle`). Apart from the numerous avenues for theoretical studies, this affords the practising engineer special challenges as well.

UI This requires a fairly simple user interface. Implementation will only have to identify and deal with mouse clicks and very little else.

Abstraction The variations of this game span different sizes of the puzzle i.e. 4x4, 5x5 and so on. Thus this is an introduction to parametrising the implementation.

Variation We will introduce a variation in this projectlet namely instead of numbered tiles, a supplied image is used. Thus solution then requires the original image to be reproduced by sliding the individual tiles according to the stated rules.

Specification

Based on the above broad ideas, the following specifications will guide the exploration.

Choice of faces The number of tiles and an image file in a standard format like png or jpg are the basic parameters to an implementation. The user should be allowed to dynamically change the tile face.

User commands The user shall be able to approach the puzzle in a different ways. Starting with a jumbled puzzle, the user can slide the tiles to produce the solution. Alternatively starting with a solution, the tiles can be manually jumbled to produce interesting patterns.

Graphic libraries Quite a few GUI libraries are available to choose from. E.g. `https://www.gtk.org/` is a platform, language agnostic choice. So is `https://www.qt.io/` though with more licensing restrictions. There are more language specific choices as in **fyne** (`https://fyne.io/`) which may be very **go** specific. **Dotnet** `https://dotnet.microsoft.com/` rounds out this list as a whole platform including user interfaces.

Implementation

Challenges

Concurrency Model The gtk library specifies that all changes to the GUI be performed from a single thread of execution. Others such as fyne do not have such an explicit specification. In this example, since every action in the puzzle is driven by the user - by mouse clicks or button presses, the design is quite simple. Except when we need to update the think time which counts up independently.

Image manipulation The need to support images as tile faces exposes us to some basic image processing needs eg to crop, recombine images. More interesting however will be the investigation of the marriage between the static image representation in files (.png or .jpg) and the more dynamic GUI environment and the widget sets presented by the library.

Mouse Clicks Mouse clicks are the way the sliding tiles are identified. Each library supports a different way of identifying mouse clicks and their position.

Algorithm Hiding In order to experiment with different GUI libraries, the algorithmic aspects need to be isolated and a clean interface established.

Example Project

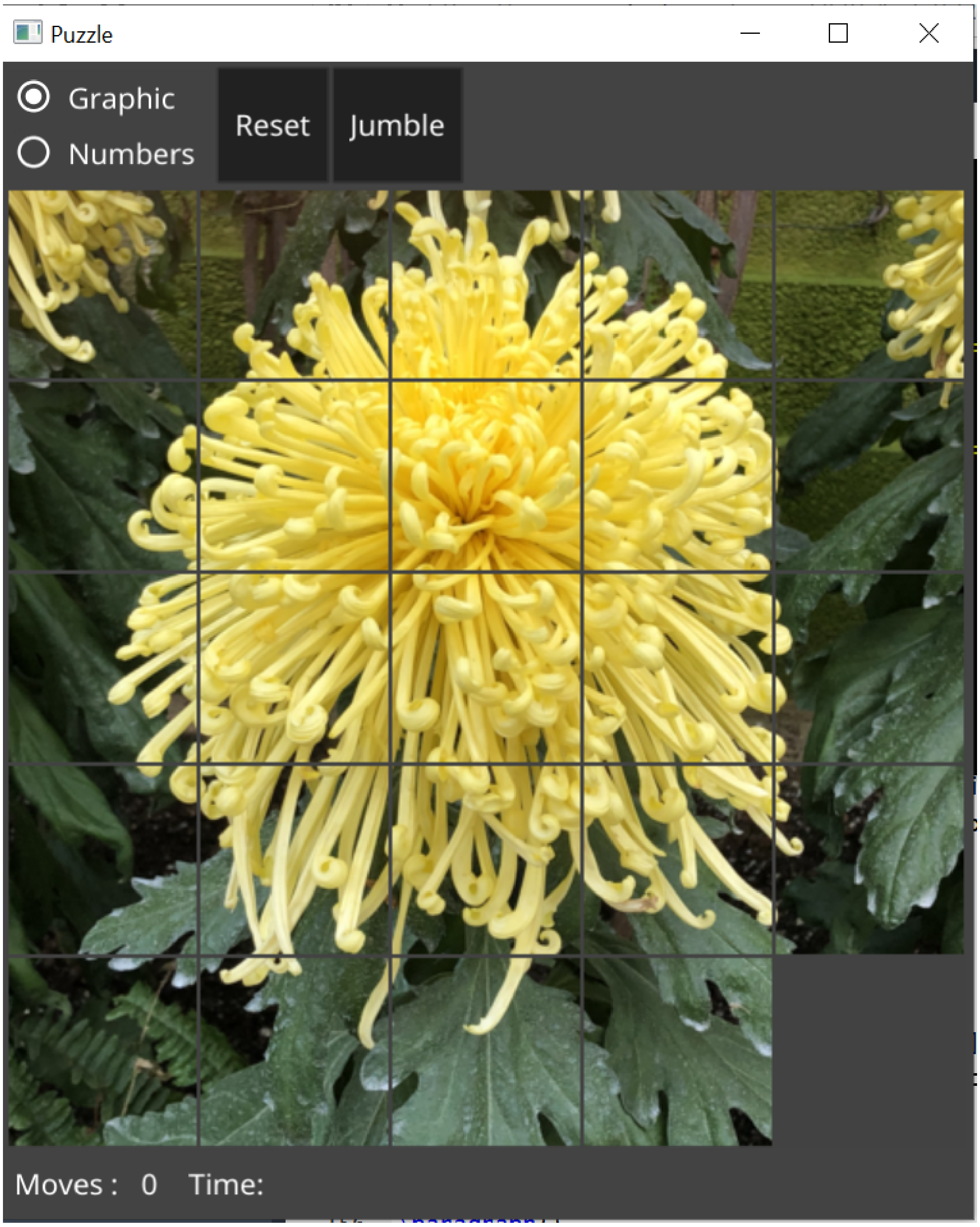
An example project has been developed to illustrate the concepts - with the following specifications:

```
usage: puzzle [no of segments] [imagename] [fyne|gtk]
use . for segments=4 as well as etc/flower.png
```

An implementation in **go** language has been developed using fyne and gotk3 bindings to gtk `https://github.com/gotk3/gotk3` and is available for exploration and enhancements.

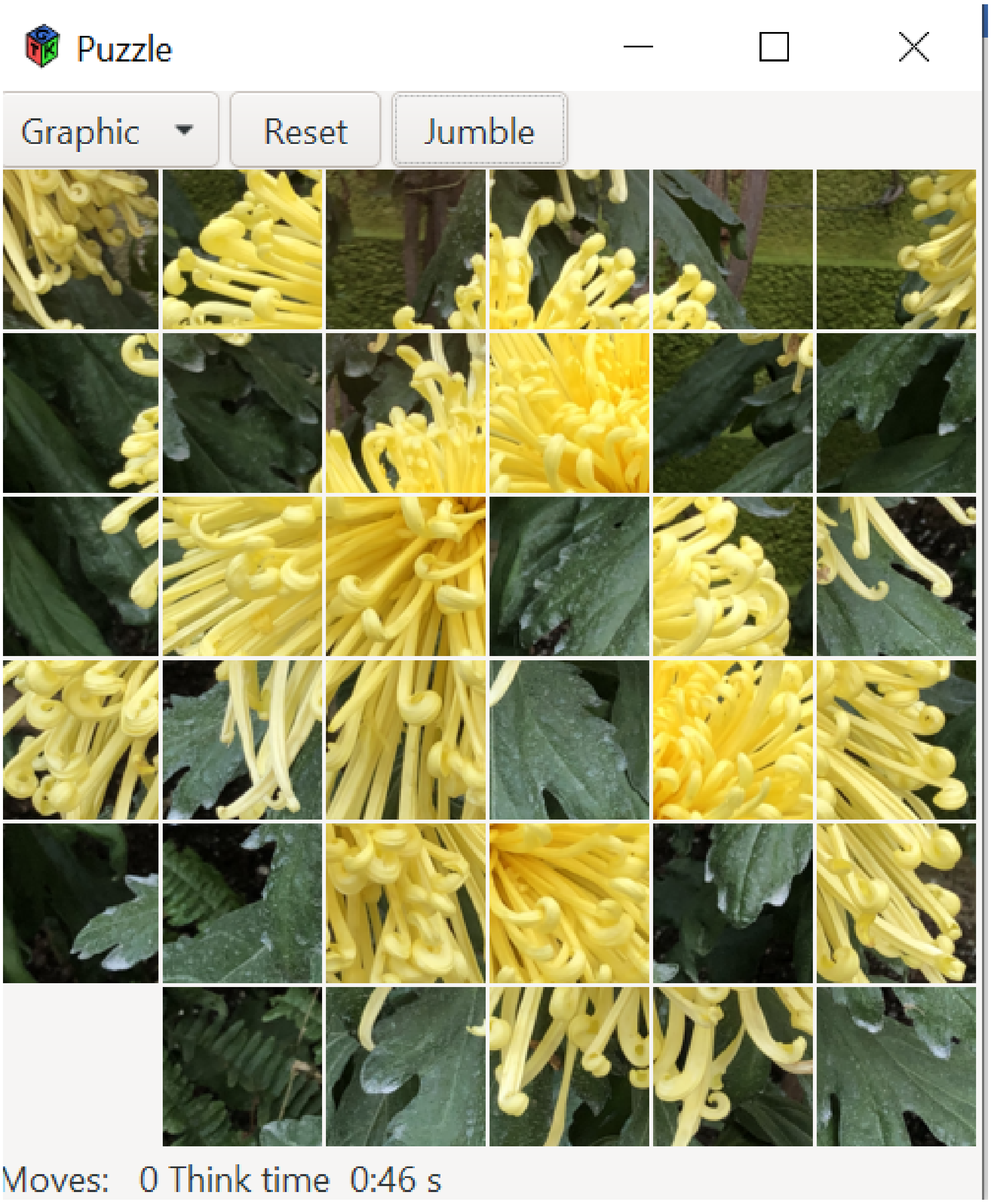
```
https://ToprLLC@bitbucket.org/ToprLLC/puzzle.git
```

Implementation with fyne graphical library



Moves : 0 Time:

Implementation with gtk - Jumbled puzzle



Moves: 0 Think time 0:46 s

Potential Improvements

Undo Keep track of each move and be able to retract.

Patterns Instead of random Jumbles, arrange them in a different predictable pattern. For instance, in the equivalent numeric view, the numbers will be placed along the periphery clockwise and so on eventually with the last empty tile being close to the center.

Solver Include a solver which finds the best solution from a given state presenting an animated view.